OPIC
OFFICE DE LA PROPRIÉTÉ INTELLECTUELLE DU CANADA

CIPO
CANADIAN INTELLECTUAL PROPERTY OFFICE

(12) (19) (CA) **Demande-Application**

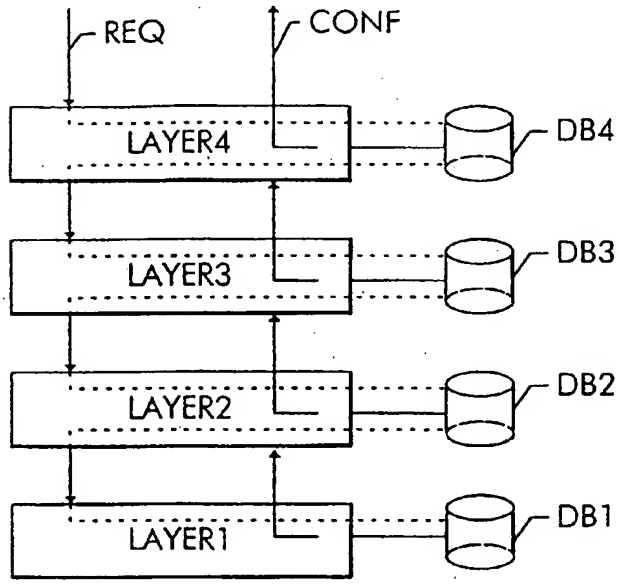
(21) (A1) **2,272,425**
(22) 1999/05/19
(43) 1999/11/20

(72) JÄKEL, Hans-Jörg, DE
(72) BANZHAF, Monika, DE
(72) KOCHER, Hartmut, DE
(71) ALCATEL, FR

(51) Int.Cl.[6] G05B 15/02
(30) 1998/05/20 (198 22 551.2) DE

(54) **SYSTEME COMMANDE PAR PROCESSEUR ET METHODE D'EXPLOITATION D'UN TEL SYSTEME**

(54) **PROCESSOR-CONTROLLED SYSTEM AND METHOD OF OPERATING A PROCESSOR-CONTROLLED SYSTEM**



(57) In a system (SYS) controlled by a processor by means of a control program, the control program consists of several hierarchically arranged functional layers (LAYER1-4). Each of the program layers provides services, at least part of the services of a higher program layer building on services of a lower program layer. The program layers are linked asynchronously, such that a request (REQ) received from a higher program layer is responded to by sending a confirmation (CONF) before the request (REQ) is forwarded to a next lower program layer. As a result, the program layers are separated with respect to the time sequence of the program run. Advantageously, each of the program layers (LAYER1-4) has a memory (DB1-4) associated with it in which configuration data of the respective program layer are stored. A request (REQ) is first processed locally in a current program layer by updating the configuration data in the memory. Then, the confirmation (CONF) is sent, whereupon the request is forwarded to the next lower program layer for further processing.

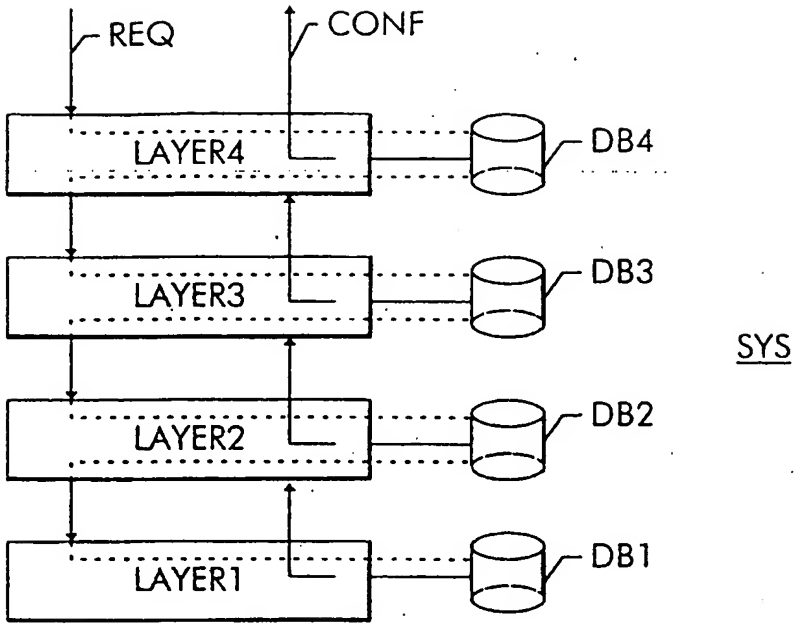

REQ
CONF
LAYER4
LAYER3
LAYER2
LAYER1
DB4
DB3
DB2
DB1
SYS

Fig.1

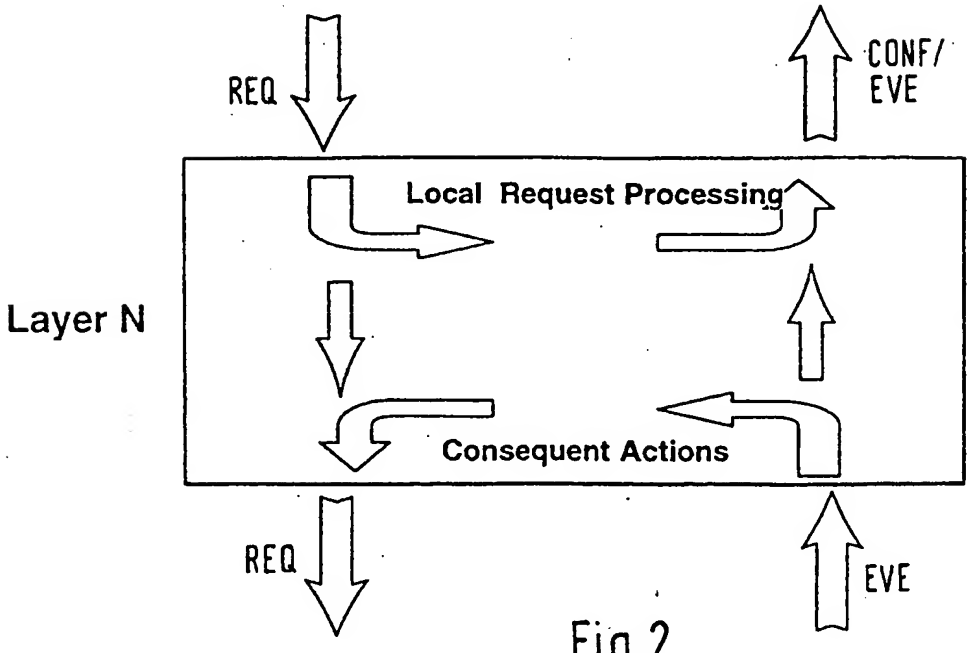

REQ
CONF/
EVE
Local Request Processing
Layer N
Consequent Actions
REQ
EVE

Fig.2

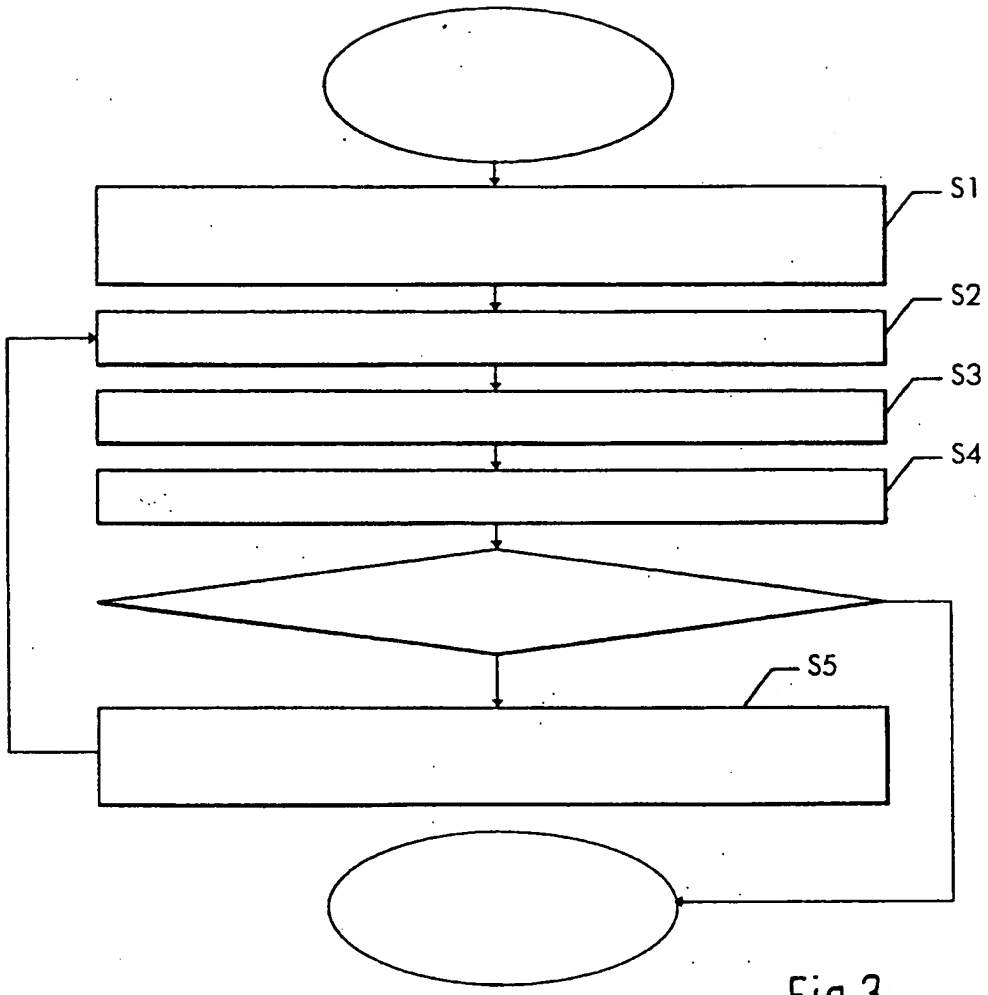


Fig.3

Processor-Controlled System and Method of Operating a Processor-Controlled System

This invention relates to a method of operating a processor-controlled system as set forth in the preamble of claim 1, particularly a network management system, and to a processor-controlled system as set forth in the preamble of claim 5.

For many applications, systems are controlled by one or more processors executing a control program, such as an operating system. For the control program, an architecture of hierarchically arranged program layers is frequently chosen, with each program layer providing different services, and at least part of the services of a higher layer building on services of a lower layer. The layered architecture is used particularly for network management systems.

In an article by M. P. Bosse et al, "Management von SDH-Netzelementen: eine Anwendung der Informationsmodellierung", Elektrisches Nachrichtenwesen, 4th Quarter 1993, pp. 329-338, the structure of a control program for a network management system of an SDH network (SDH = synchronous digital hierarchy) is described. The control program consists of different, hierarchically arranged program

layers. These program layers comprise a network layer and an element layer. The article also mentions that the network management system has database capabilities. Communication between the program layers takes place via a defined interface referred to as a "Q3 interface".

According to an article by S. Colombo, "Technologie der SDH-Netzelemente", Elektrisches Nachrichtenwesen, 4th Quarter 1993, pp. 322-328, the element layer is also a control program with a layered architecture. The network element functions reside in a first program layer in the form of managed objects. This program layer builds on functions which are provided by the program layer referred to as "virtual hardware module". Located below the virtual hardware module is the on-board controller software, which accesses the SDH hardware. The first program layer incorporates a permanent ("persistent") storage for storing configuration parameters (managed objects) in a database referred to as a "persistent database".

With the layered architecture described, communication between adjacent program layers is necessary: A higher program layer must forward requests to a lower program layer, and the lower program layer must return a confirmation of the execution of the request and, if necessary, a result. A higher program layer generally waits for the confirmation, and the lower program layer does not send the confirmation until the request has been successfully executed.

A problem associated with this approach is that with a

layered program architecture, the entire system may be blocked when a higher program layer is waiting for a confirmation from a lower program layer when the lower program layer is busy and does not get around to executing the request received from the higher program layer. The throughput through such a multilayer control program is thus limited by the speed of the slowest layer. Separating ("decoupling") the program layers by means of queues does not result in a basic improvement, since queues may overflow, for example in the event of a failure of a program layer due to an error.

It is an object of the invention to provide a method of operating a processor-controlled system, particularly a network management system, as well as a processor-controlled system wherein blocking cannot occur as a result of a busy condition or failure of a program layer.

This object is attained with respect to the method by the features of claim 1 and with respect to the system by the features of claim 5. Further advantageous aspects of the invention are defined in the dependent claims.

One advantage of the invention is that in the event of an error, the availability, robustness, and survivability of a system according to the invention is improved.

Another advantage is that the efficiency of a system according to the invention is improved, since several

requests can be combined and processed together in one program layer. In addition, throughput through the individual program layers is not limited by the processing speed of the slowest program layer.

A further advantage is that the system according to the invention remains operable at least externally even in the event of a total failure of a middle or lower program layer. A special case of a total failure exists if part of the system is shut down or not connected. The invention thus permits a transparent off-line configuration of systems according to the invention.

A preferred application of the invention is in the area of network management, namely as a network management system of a telecommunications network or as a controller of a network element of a telecommunications network.

The invention will become more apparent from the following description of an embodiment when taken in conjunction with the accompanying drawings, in which:

Fig. 1 shows schematically the structure of a system in accordance with the invention;
Fig. 2 shows the data flow through a program layer of a system in accordance with the invention; and
Fig. 3 is a flowchart of the method in accordance with the invention.

To separate ("decouple") individual layers of a

control program with respect to the time sequence of the program run, according to the invention an asynchronous approach is chosen. A basic idea of the invention is to first process a request locally in a higher program layer and send back a confirmation, and subsequently forward the request to a lower program layer for further processing. This imparts to the confirmation the character of a promise to take care that the request will be executed. The confirmation thus no longer states that the request was successfully processed, but acknowledges the receipt of the request and promises to process it. By these measures, effective, complete decoupling of the program layers is achieved.

According to the invention, a user, in response to a request, receives a confirmation in the form of an "OK" before the request was actually executed. Particularly advantageously, the request is logically and semantically checked in a current program layer. This ensures that the request is executable at least in principle, and the confirmation then represents an assurance that the request will be executed.

In order that local processing of the request can be performed, each of the program layers has a memory associated with it in which the current configuration parameters of the respective layer are stored and saved. "Configuration parameters" as used herein also means event and status data of the respective program layer. The memories of the program layers are logically separate memories which, however, may be physically implemented in the same memory device, such

as a RAM, an EEPROM, a hard disk, or another data carrier. Preferably, the memory is structured as a database in which the configuration parameters are stored. The control program may also be complex control software consisting of a number of program modules which are executed as a distributed application on different processors.

In the embodiment shown in Fig. 1, the system according to the invention, SYS, consists of four program layers LAYER1 to LAYER4. Each of the program layers has a database DB1 to DB4 associated with it. The databases contain configuration parameters of the associated program layer. To obtain access to the system SYS, a request REQ is sent to the highest program layer LAYER4. In this program layer LAYER4, local processing of the request REQ is performed by updating the configuration parameters affected by the request in the database DB4 in accordance with the request. When this has been done, a confirmation CONF is sent in response to the request REQ.

If the request REQ needs to be processed in the next lower program layer LAYER3, it will be forwarded to this layer. There, the same steps are taken: local processing of the request by updating the affected configuration parameters in the database DB3, and subsequently sending a confirmation.

The forwarding to the respective next lower program layer continues until the lowest program layer necessary to execute the request is reached.

This will now be illustrated by the example of a network management system. The program layer LAYER4 is an application layer in a central network management facility. The next lower program layer LAYER3 is the so-called Management Information Base (MIB, as standardized in ITU-T X.720) together with a software "framework" in a controller of a network element, for example of a digital crossconnect. The next lower program layer LAYER2 is a virtual hardware module VHM which performs the conversion between the MIB and the hardware. The lowest program layer LAYER1 is the firmware, which is held on the individual boards of the crossconnect and controls the so-called on-board controllers (OBC).

A request from a user to change a particular connection parameter of an existing connection must be processed by all four program layers. By contrast, a request to read a connection parameter of an existing connection can already be fulfilled by the first or second program layer, depending on the type of the parameter.

A special advantage of the invention is that several requests can be combined and processed together in one program layer. If, for example, the highest program layer receives several requests which all want to change, directly or indirectly, a particular parameter, then the particular parameter will actually be set to a new value in the highest program layer several times in succession. This, however, takes place only locally. When being forwarded, the several requests can be combined into a single one, so that in

each of the lower layers, only a single change needs to be made to the respective parameter. This approach is particularly advantageous in case of an off-line configuration of the system according to the invention, i.e., if the lower program layers are not available. In that case, all changes will be made in the local database, and after system start-up, the changes necessary as a result of the off-line configuration will be forwarded in a single request to the lower program layers.

Event reports can initiate consequent actions at the next lower layer. This results in a further advantage of the invention, namely that a request to perform the consequent action can be sent to the next lower layer even if the next higher layer, to which this event report is to be forwarded, cannot currently process or accept these event reports. Several event reports can be combined if the higher program layer to which they are to be forwarded cannot accept any reports for a prolonged period of time. In this manner, only the respective current status is forwarded. Events are thus processed locally and, if possible, are combined when being forwarded (adaptive event suppression).

Fig. 2 shows schematically the data flow in a program layer. The program layer LAYER N receives requests REQ from the next higher program layer, processes the request locally, and sends as a response a confirmation CONF. The request REQ is then forwarded to the next lower program layer. In the reverse direction, the program layer LAYER N receives event reports EVE from the next lower program layer. Certain

events may require consequent actions, so that processing of the event report is performed in the program layer LAYER N by sending to the next lower program layer a request REQ to perform the consequent action. In response to the event reports, the internal database is updated if necessary. The event report EVE is then forwarded to the next higher program layer.

Fig. 3 shows a flowchart of the method according to the invention. The method comprises the following steps:

Step S1: The system is accessed by sending a request to the highest program layer. The highest program layer thus becomes the current program layer.

Step S2: The request is executed in the current program layer.

Step S3: The configuration data in the memory, preferably in the database, are changed in accordance with the request.

Step S4: As a response, a confirmation is sent.

Step S5: If the lowest program layer needed to execute the request has not yet been reached, the request will be forwarded to the next lower program layer for further processing. The next lower program layer thus becomes the current program layer. Then, steps S2 to S5 are repeated.

"Current program layer" as used herein means the program layer which is currently processing the request locally.